REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	951101	Technical

4. TITLE AND SUBTITLE

Godunov Methods for Nonlinear Solids

5. FUNDING NUMBERS

C - DNA 001-92-C-0166
PE - 62715H
PR - AC
TA - BI
WU - DH327280

6. AUTHOR(S)

John A. Trangenstein

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Duke University
Department of Mathematics
P. O. Box 90320
Durham, NC 27708-0320

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Nuclear Agency
6801 Telegraph Road
Alexandria, VA 22310-3398

TAIC/Rohr

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

DNA-TR-94-143

11. SUPPLEMENTARY NOTES

This work was sponsored by the Defense Nuclear Agency under RDT&E RMC Code B 4662 D AC BI 00011 7010 A 25904D.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The adaptive mesh refinement algorithm for unsteady gas dynamics is extended to nonlinear solids. Several modifications to the original algorithm are forced by the dissimilarities in the forms of the gas and solid equations of state. The variety of forms of the solid constitutive laws motivates the development of several abstractions that are implemented through object-oriented programming. Examples of the performance of the scheme are provided by two numerical examples.

14. SUBJECT TERMS

Adaptive Mesh Refinement
Solid Mechanics
Wave Propagation
Finite Differences
Hyperbolic Conservation Laws

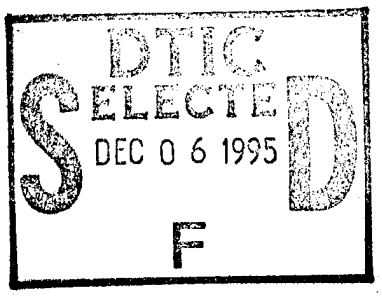
15. NUMBER OF PAGES

26

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

Defense Nuclear Agency
Alexandria, VA 22310-3398



DNA-TR-94-143

Godunov Methods for Nonlinear Solids

John A. Trangenstein
Duke University
Department of Mathematics
P.O. Box 90320
Durham, NC 27708-0320

November 1995

Technical Report

CONTRACT No. DNA 001-92-C-0166

Destroy this report when it is no longer needed. Do not return to sender.

PLEASE NOTIFY THE DEFENSE NUCLEAR AGENCY, ATTN: CSTI, 6801 TELEGRAPH ROAD, ALEXANDRIA, VA 22310-3398, IF YOUR ADDRESS IS INCORRECT, IF YOU WISH IT DELETED FROM THE DISTRIBUTION LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION.

DISTRIBUTION LIST UPDATE

This mailer is provided to enable DNA to maintain current distribution lists for reports. (We would appreciate your providing the requested information.)

☐ Add the individual listed to your distribution list.

☐ Delete the cited organization/individual.

☐ Change of address.

NOTE:
Please return the mailing label from the document so that any additions, changes, corrections or deletions can be made easily. For distribution cancellation or more information call DNA/IMAS (703) 325-1036.

NAME: _____

ORGANIZATION: _____

OLD ADDRESS	CURRENT ADDRESS
_____	_____
_____	_____
_____	_____

TELEPHONE NUMBER: ( ) _____

DNA PUBLICATION NUMBER/TITLE	CHANGES/DELETIONS/ADDITIONS, etc.) *(Attach Sheet if more Space is Required)*
_____	_____
_____	_____
_____	_____

DNA OR OTHER GOVERNMENT CONTRACT NUMBER: _____

CERTIFICATION OF NEED-TO-KNOW BY GOVERNMENT SPONSOR (if other than DNA):

SPONSORING ORGANIZATION: _____

CONTRACTING OFFICER OR REPRESENTATIVE: _____

SIGNATURE: _____

CUT HERE AND RETURN

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	951101	Technical

4. TITLE AND SUBTITLE

Godunov Methods for Nonlinear Solids

5. FUNDING NUMBERS

C - DNA 001-92-C-0166
PE - 62715H
PR - AC
TA - BI
WU - DH327280

6. AUTHOR(S)

John A. Trangenstein

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Duke University
Department of Mathematics
P. O. Box 90320
Durham, NC 27708-0320

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Nuclear Agency
6801 Telegraph Road
Alexandria, VA 22310-3398

TAIC/Rohr

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

DNA-TR-94-143

11. SUPPLEMENTARY NOTES

This work was sponsored by the Defense Nuclear Agency under RDT&E RMC Code B 4662 D AC BI 00011 7010 A 25904D.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The adaptive mesh refinement algorithm for unsteady gas dynamics is extended to nonlinear solids. Several modifications to the original algorithm are forced by the dissimilarities in the forms of the gas and solid equations of state. The variety of forms of the solid constitutive laws motivates the development of several abstractions that are implemented through object-oriented programming. Examples of the performance of the scheme are provided by two numerical examples.

14. SUBJECT TERMS

Adaptive Mesh Refinement
Solid Mechanics
Wave Propagation
Finite Differences
Hyperbolic Conservation Laws

15. NUMBER OF PAGES 26

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

CLASSIFIED BY:
N/A since Unclassified.

DECLASSIFY ON:
N/A since Unclassified.

SIAM J. SCI. COMPUT.
Vol. 16, No. 4, pp. 819–839, July 1995

004

ADAPTIVE MESH REFINEMENT FOR WAVE PROPAGATION IN NONLINEAR SOLIDS*

JOHN A. TRANGENSTEIN†

Abstract. The adaptive mesh refinement algorithm for unsteady gas dynamics is extended to nonlinear solids. Several modifications to the original algorithm are forced by the dissimilarities in the forms of the gas and solid equations of state. The variety of forms of the solid constitutive laws motivates the development of several abstractions that are implemented through object-oriented programming. Examples of the performance of the scheme are provided by two numerical examples.

Key words. adaptive mesh refinement, solid mechanics, wave propagation, finite differences, hyperbolic conservation laws

AMS subject classifications. 35L65, 39-04, 65M50, 65N50, 73D15, 73V15

1. Introduction. Wave propagation in nonlinear solids involves localized phenomena that is difficult to capture with standard shock-capturing techniques. For example, shear-band formation, crack propagation, and wave refraction at material interfaces involve highly localized and complicated wave structures. If these waves are modeled using a regular mesh, so that sufficient refinement is available where it is needed, then a significant amount of the computational work is performed unnecessarily in regions away from the waves of interest. The mesh refinement can be reduced through the use of high-order difference methods; however, this still does not concentrate the computational work where it is needed most.

Efficient treatment of nonlinear wave propagation can be achieved through the combination of accurate shock-capturing methods with adaptive mesh refinement. Our approach to the selection of the shock-capturing method has been to adapt successful high-resolution gas dynamics schemes to solid mechanics. We chose a second-order variant of Godunov's method because of its success in resolving interesting gas dynamic flows [19]. We have conducted a comparison of the dissipation and dispersion of the one-dimensional Godunov method for solid mechanics with standard explicit finite element techniques in one dimension [15], and have shown that Godunov's method achieves greater resolution of both rarefactions and shocks with no user-controlled parameters (other than timestep selection). We have also described a version of Godunov's method for two-dimensional solids [16], following the ideas in [3]. The purpose of this paper is to describe the combination of adaptive mesh refinement with the Godunov algorithm for solids.

By adaptive mesh refinement, we mean the selection of nested lists of rectangular patches of refined computational cells, following the ideas in [2]. The principal advantage of adaptive mesh refinement is that it can generally obtain a desired level of accuracy with nearly minimal computational cost. An alternative approach [10] is to try to obtain maximum accuracy for a fixed cost. Other approaches use variable numbers of computational cells on unstructured meshes or meshes refined cell-by-cell [9].

There are both advantages and disadvantages to the form of adaptive mesh refinement employed in this paper. The principal difficulty is the programming complexity. The computations are carried out on lists of logically rectangular patches defined in recursively finer index spaces, with each list designed so that the union of its patches is contained in the union of the

*Received by the editors September 14, 1993; accepted for publication (in revised form) May 31, 1994.

†Department of Mathematics, Duke University, Durham, North Carolina 27708-0320 (johnt@math.duke.edu). This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract W-7405-Eng-48. Partial support was provided by the Applied Mathematical Sciences Program of the Office of Energy Research and by the Defense Nuclear Agency.

next coarser list of patches. A fair amount of programming is required to handle the communication between the patches on the same and different levels of refinement. The intent in this design is to generate a small number of rich computational assignments (i.e., integrate a patch) using a highly structured numerical method. One rationale behind this design principle is that much more is known about the design of accurate and efficient numerical shock-capturing methods on logically rectangular meshes. It is also possible to make efficient use of vector and parallel processors.

It is difficult to predict the relative success of adaptive mesh refinement in comparison to the use of very high order methods on a fixed mesh. The experience in the development of adaptive integrators for ordinary differential equations has been that it is more efficient to refine the mesh and drop the order of the scheme near rough behavior [6], [8]. The experience in gas dynamics has been that it is difficult to construct high *accuracy* (as opposed to *order*) schemes near strong discontinuities. In particular, it is possible that a higher-order scheme is actually less accurate than a low-order scheme in such a case. This is because the local truncation error can be thought of as the mesh width (or timestep) raised to some power, times a high-order derivative of the solution to the equation. As the order of the scheme increases, the leading error term could show a rate of growth in the size of the derivative of the solution that is greater than the corresponding power of the mesh size or timestep.

We have other motivations in applying adaptive mesh refinement to solids. In applications involving plasticity, the equation of state typically is described by a system of constrained ordinary differential equations. Controlling the accuracy of the integration of the elastic-plastic equation of state appears to be the key to controlling the accuracy of the overall scheme. Thus a natural strategy is to refine the mesh in places where the stress integration detects significant error. We also note that very complicated wave behavior can be expected in nonlinear solid mechanics [4], [7], [11], [13], [17], including local discontinuities in the characteristic speeds and directions, local linear degeneracies, loss of strict hyperbolicity, change of type, and cusping and interaction of the characteristic cones. Therefore, high-order schemes should be used with care on such problems.

The rest of the paper will present the details of our approach. First, we will briefly review the equations of motion for solids and Godunov's method for their solution. In §3, we will present an overview of the adaptive mesh refinement algorithm. In the fourth section, we will discuss some of the special features required by solid mechanics computations. Afterward, we will discuss how certain features of object-oriented programming have made the algorithm easier to implement. We will conclude the paper with some numerical examples.

2. Godunov's method for solids. In order to describe the Lagrangian equations of motion for deformation of a two-dimensional isothermal solid, we need to introduce some notation. Let ρ be the density at rest, v be the velocity, F be the deformation gradient, S be the first Piola–Kirchhoff stress tensor, t be time, and x be position in the original (Lagrangian) configuration. The equations of motion can be written

$$\frac{\partial u^{\mathsf{T}}}{\partial t} + \nabla_x^{\mathsf{T}} G^{\mathsf{T}} = 0. \tag{1}$$

where the vectors of conserved quantities and fluxes are

$$u = \begin{bmatrix} \mathrm{v}\rho \\ Fe_1 \\ Fe_2 \end{bmatrix}, \qquad G = \begin{bmatrix} -FS \\ -\mathrm{v}e_1^{\mathsf{T}} \\ -\mathrm{v}e_2^{\mathsf{T}} \end{bmatrix}. \tag{2}$$

Here, e_1 and e_2 are the unit axis vectors in the respective coordinate directions.

In addition to these conservation laws, we also require an equation of state relating the stress (or rate of stress) to the deformation gradient (or rate of deformation) in an appropriate rotationally invariant fashion. For the purposes of this paper, we shall use either of two models. The first is linear elasticity:

$$\frac{dS}{dt} = \left(\frac{dF}{dt} + \frac{dF^\top}{dt}\right)\mu + I\lambda \mathrm{tr}\,\frac{dF}{dt}.$$

Here, $\lambda = \kappa - \frac{2}{3}\mu$ is a Lame constant, κ is the bulk modulus, and μ is the shear modulus. The second model we will use is a hyperelastic model

$$S = 2F\frac{\partial \rho\psi}{\partial C},$$

where the Helmholtz free energy per unit mass ψ is given by the Mooney–Rivlin model [12]

$$\rho\psi(C) = \lambda\nu(|\,C\,|^{\frac{1}{3}}) + \frac{\mu}{2}\mathrm{tr}\,C - \frac{\mu}{2}\ln|\,C\,|.$$

Here $C = F^\top F$ is the Green deformation tensor, $|\,C\,|$ is its determinant, and

$$\nu(\gamma) \equiv \frac{1}{2}(\ln\gamma)^2.$$

Our Godunov algorithm for the solution of (1) is based on the following conservative difference on a rectangular mesh:

$$u_{ij}^{n+1} = u_{ij}^n - (G_{i+\frac{1}{2},j}^{n+\frac{1}{2}} - G_{i-\frac{1}{2},j}^{n+\frac{1}{2}})e_1\frac{\Delta t}{\Delta x_1} - (G_{i,j+\frac{1}{2}}^{n+\frac{1}{2}} - G_{i,j-\frac{1}{2}}^{n+\frac{1}{2}})e_2\frac{\Delta t}{\Delta x_2}. \tag{3}$$

Here, i and j are the indices of cell centers, and n is the timestep number. For a hyperelastic equation of state, such as the Mooney–Rivlin model being employed in this paper, it is a simple matter to evaluate the stress at the new timelevel after the conservative difference. For a hypoelastic equation of state, it is necessary to integrate the equation of state along particle paths.

The computation of the conservative fluxes $G_{i\pm\frac{1}{2},j}^{n+\frac{1}{2}}$ and $G_{i,j\pm\frac{1}{2}}^{n+\frac{1}{2}}$ involves several steps. There is the construction of a conservative piecewise linear interpolant to the flux variables, characteristic tracing and transverse flux correction to estimate states describing the interaction between cells, and the approximate solution of Riemann problems to determine the numerical fluxes. For details, the reader can consult [3], [16].

For the purposes of this paper, there are several features of this numerical scheme to remember. First, the conserved quantities (velocity v and deformation gradient F) are located at cell centers, and therefore are well defined on an adaptively refined mesh. Second, the fluxes G are associated with the centers of cell space-time sides, presenting some ambiguity that must be resolved during adaptive mesh refinement. Third, the construction of conservative piecewise linear interpolants for the flux variables within each cell involves information on neighboring cells: thus in order to obtain second-order accuracy up to the boundary of a patch it is necessary to obtain information on "ghost cells" around the outside of the boundary of the patch. Fourth, the stress at a cell center may be computed from the deformation gradient (hyperelastic models) or by integrating the rate of deformation along particle paths (hypoelastic models).

3. Overview of adaptive mesh refinement.

3.1. Basic assumptions. Now that we have presented the important features of the finite difference method, we turn to a description of adaptive mesh refinement. We shall begin by describing the steps in integrating a list of patches at some level of mesh refinement. Later, we shall describe how these lists are generated initially.

There are several assumptions built into our adaptive mesh refinement scheme. We assume that we are given lists of logically rectangular patches corresponding to the levels of refinement of the coarsest mesh; we shall concentrate our discussion on the integration of the patches in one of these lists. We are also given some time increment through which we must integrate the data on this list of patches. We assume that the boundaries of patches on the next finer level always align with the boundaries of cells on this level of refinement. In other words, when a cell is marked for refinement, all of it is refined. Furthermore, we assume that the union of patches on a finer level of refinement is contained in the interior of the union of patches on the current level of refinement. Next, we assume that after the list of patches is advanced one timestep, the underlying fine patches are integrated in time until they are synchronized with the patches on this level. We further assume that regridding of finer levels is performed precisely every "regrid interval" timestep on this level; the regrid interval is either 2 or 3, depending on the kinds of variables that occur in the model. Thus error estimation is performed on all but the finest level, and only at fixed intervals in the number of timesteps (i.e., not every timestep).

3.2. Outline of the algorithm. The first task in integrating the list of patches on a given level of refinement is to estimate a stable timestep size and to reduce it if necessary so that the correct number of timesteps is taken to reach synchronization with a coarser level (if it exists). If no further refinement of the patches on this level is allowed, then there is no restriction on the number of timesteps other than the Courant–Friedrichs–Lewy (CFL) condition for numerical stability. If this level allows refinement, then it is necessary to make sure that the number of timesteps will be a multiple of the regrid interval. Unlike [2] we do not assume that a successful timestep on a coarse level can be divided by the spatial refinement ratio to provide a successful timestep on a fine level. It has been our experience that fine grid calculations detect the effects of material hardening and the resulting increase in characteristic speeds much better than coarse grids. We do attempt to make sure that roughly the same size timestep is used in each step on a given level until synchronization with a coarser level occurs.

After determining the initial timestep, we loop over timesteps until synchronization occurs. Since shock loading of nonlinear solids can lead to significant increases in the characteristic speeds, for example due to a hardening of the material as internal voids are compressed, we allow the timestep to be adjusted during this loop.

The first task in the timestep loop is to advance the patches on this level of refinement through the chosen time increment. Note that boundary data must be provided for each patch before it can be integrated; this is described in §3.3. If the patches on this level can be refined, it is useful to store the numerical fluxes used to update the conserved quantities, so that we can correct the integration of hypoelastic equations of state after integration on nearby fine cells. If this level is not the coarsest level of patches, it is also necessary to keep track of the time integral of the fluxes around the boundaries of the patches, so that we may correct the integration of the coarser patches later. The need for the storage of the time integral of the boundary fluxes is described in §3.4.

The next step in the timestep loop is to advance the finer levels of patches up to the new time. This involves a recursive invocation of the algorithm being described. In particular, we note that the integration of the next finer level of patches can change the data on the current level of patches. The way in which this occurs will be described in §3.4. Figure 1 shows the order of the timesteps on three levels of mesh refinement, illustrated by timelines.

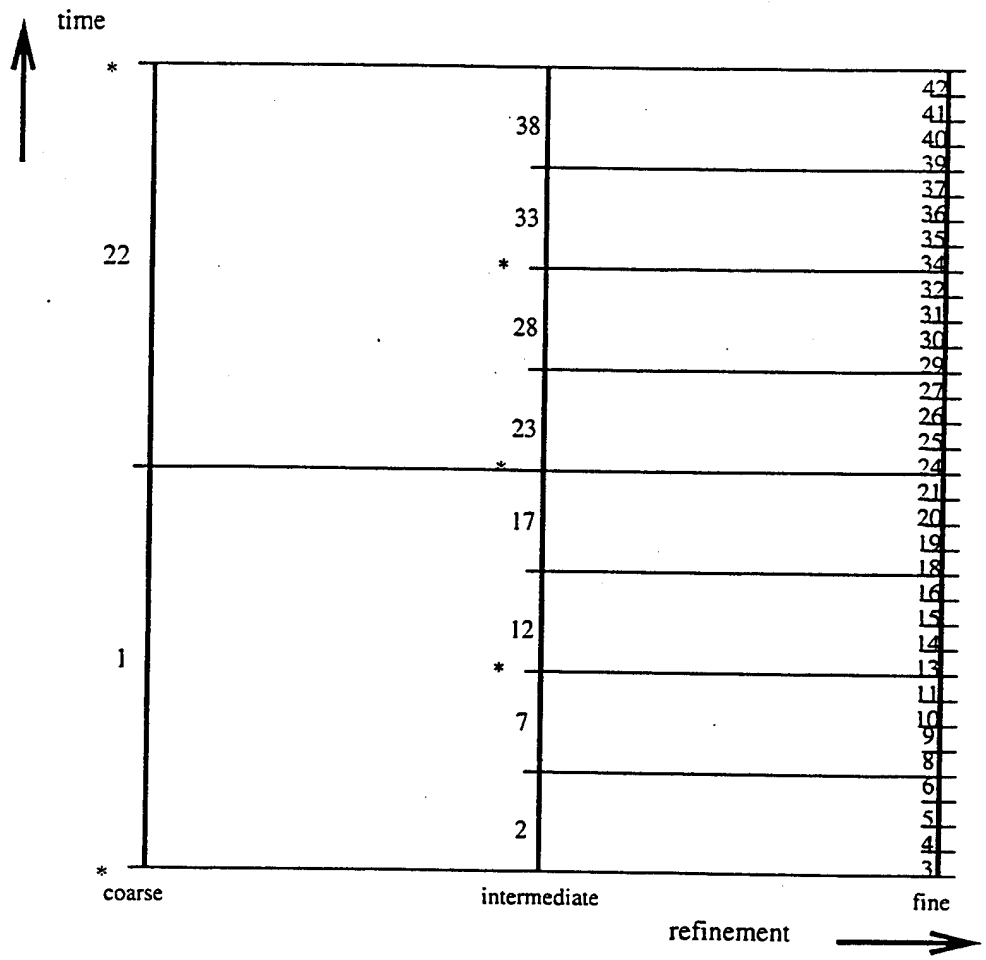


FIG. 1. *Timestep sequence during adaptive mesh refinement with refinement ratio = 4 and regrid interval = 2. Asterisks mark the timesteps and levels where error estimation is initiated.*

The third step in the loop over this list of patches is to regrid the finer levels. Regridding is performed only when the number of timesteps taken in the loop over timesteps on the current level of refinement is an integer multiple of the regrid interval; the timestep selection described above guarantees that the total number of timesteps on any level with underlying refinement will be an integer multiple of the regrid interval. If the timestep is the last in the loop, regridding may be deferred until the recursive call from a coarser level of patches. This regridding step will be described in §3.4. The reader can examine the timelines in Fig. 1; there the regridding steps are indicated by an asterisk next to the level and time at which the regridding of finer levels is initiated, after all finer levels have been advanced to that time.

When the timestep loop is complete, it is necessary to correct the data on coarser levels. This is described in §3.5. This description of the steps in the program has been brief; the difficult details are contained in the four steps that remain to be described.

3.3. Boundary data. Coarse patches are integrated before fine patches; thus fine patches cannot provide boundary data for coarser patches. Instead, boundary data are first sought from the user-prescribed boundary data at the edge of the computational domain, and then from the interior of other patches on the current level of refinement. If some boundary data cannot

be obtained in this way, then they are recursively sought from coarser levels, until all of the boundary data are provided.

Note that when data are provided by coarser levels, then that data must also be refined appropriately for the current level. For the conserved quantities (such as velocity), this means that some form of conservative interpolation is used. This interpolant has been chosen to be the same as that used in the slope construction step of the second-order Godunov method. The result is a piecewise linear function (in two dimensions) with constant term equal to the cell-centered value of the conserved quantity, and slopes modified to prevent new extrema along the coordinate directions. For other quantities (such as stress), other kinds of refinement strategies may be more appropriate. The issues of coarsening and refining data between levels will be discussed in greater detail in §4.

The strategy used to select the placement of patches in the regridding step can affect the determination of the boundary data. In particular, if we were to require that fine patches be contained within a suitable number of cells from the boundary of the union of the coarser level of patches, then recursion could be avoided in finding the boundary data (because all boundary data could be determined from the current and the next coarser level). It has been our assumption in designing the algorithm that overall efficiency would be increased most by allowing each union of patches to be only as large as necessary to contain the cells where unacceptable errors are detected; it could lead to too much computational work to enlarge these patches to contain extra cells for fine level boundary data as well.

Figure 2 illustrates the determination of boundary data for a patch. The patch of interest sits between the boundary of the physical domain (the heavy vertical line to the left) and another patch on the same level of refinement. Boundary data must be found for extra cells, called "ghost cells" (in this case four) around the outside of the patch; the boundary of the ghost cells is drawn as a thin solid line around the patch. Part of this boundary data is determined by the boundary conditions, and part is provided by the data on the other patch. The remainder of the boundary data must be obtained by refining data on the overlying coarse patches, which are illustrated by dashed lines where no fine patches occur. Although Fig. 2 indicates that the remaining boundary data can be provided by the coarser level of patches, in general it is possible that some of this refined boundary data could come from even coarser levels.

We assume that the patches have been placed so that the physical boundary does not occur in the middle of the "ghost cells." When error estimation would otherwise cause this to occur, we extend the refined region over to the physical boundary. This affects the regridding process, which is described next.

3.4. Regridding. Because we are interested in solving time-dependent problems, we must allow the mesh refinement to move in time. There are several design principles in this process. First, we want to locate the new fine mesh only where it is needed; we shall use an error estimation procedure (described below) to determine where the unacceptably large errors occur on this level. By using an error estimator, rather than a gradient detector, we are able to place mesh refinement where discontinuities in the flow variables are about to form or where the algorithm is not able to produce second-order accuracy for some other reason such as a lack of smoothness in the equation of state. Second, we want to maintain proper nesting of the levels of refinement, so that the union of patches on each level of refinement is contained in the union of patches on the next coarser level. Third, we do not want to move the mesh every timestep on each level; instead we shall move the mesh after a fixed number of timesteps, the regrid interval, in order to keep the cost of error estimation within acceptable limits. Since we are not moving the mesh every timestep, it is necessary to provide a buffer region around the cells currently requiring refinement, so that the waves of interest cannot move off the refined mesh before the next regridding step. Here, we take advantage of the

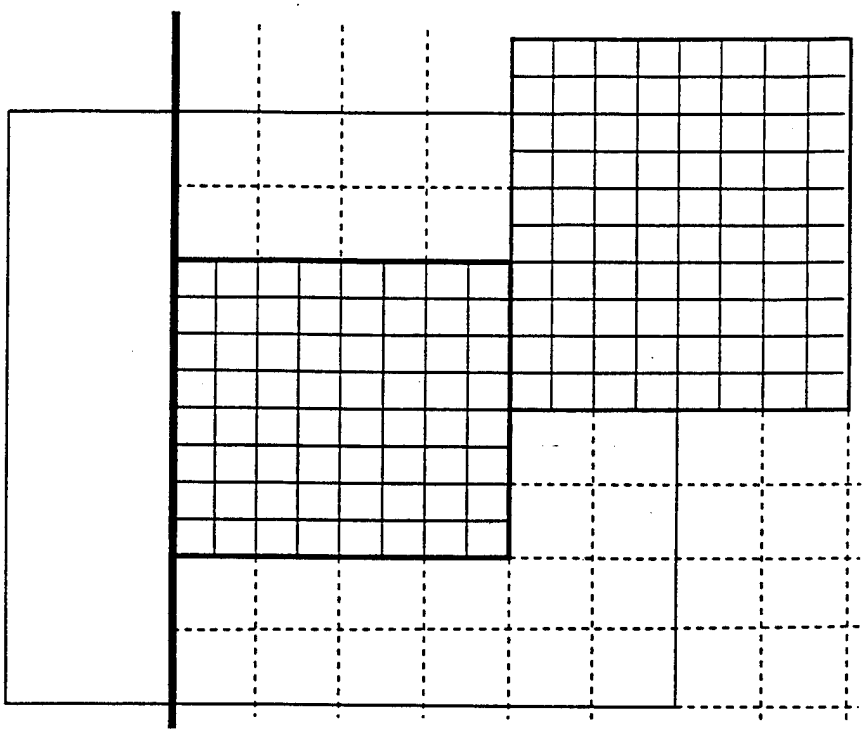


FIG. 2. *Sources of boundary data during adaptive mesh refinement. The physical boundary is the very thick vertical line at left; the fine patch currently seeking boundary data is next to the physical boundary and has a boundary represented by lines of intermediate thickness. The fine patch requires data in "ghost cells" inside a larger rectangle (two coarse cells wider in both directions) surrounding the patch, represented by solid lines both inside and outside the physical domain.*

CFL condition: the explicit integration method we are using is designed so that a wave can travel across at most one cell in a timestep. In the remainder of this section, we will discuss these ideas in greater detail.

The regridding process begins by determining a proper nesting list on the finest level that is not currently subject to mesh motion. (These are illustrated by the asterisks in Fig. 1.) This step consists of determining a list of "boxes" that completely cover the interior of the patches, where "interior" means all cells that are not within a fixed number (the "proper nesting buffer") of cells from the boundary of the union of the patches. (See §5 for the distinction between "boxes" and "patches.") We have taken the "proper nesting buffer" to be 1; this allows us to fill the data on the interior of patches on the next finer level by interpolation from the data on this level. The proper nesting list is usually computed by determining a list of boxes that exactly cover the complement of the union of the patches. Next, each box in this list is replaced by another box that is larger in each coordinate direction by the "proper nesting buffer" number of cells. Then the proper nesting list is taken to be the complement of the list of enlarged boxes in the complement list. After the proper nesting list is determined on the finest level that is not subject to mesh motion, proper nesting lists on finer levels are determined recursively. Each proper nesting list then lies the "proper nesting buffer" number of fine cells within the refinement of the proper nesting list on the next coarser level.

One important aspect of the use of the proper nesting list is that adjacent cells can differ by at most one refinement level; this is enforced by the use of the "proper nesting buffer" described

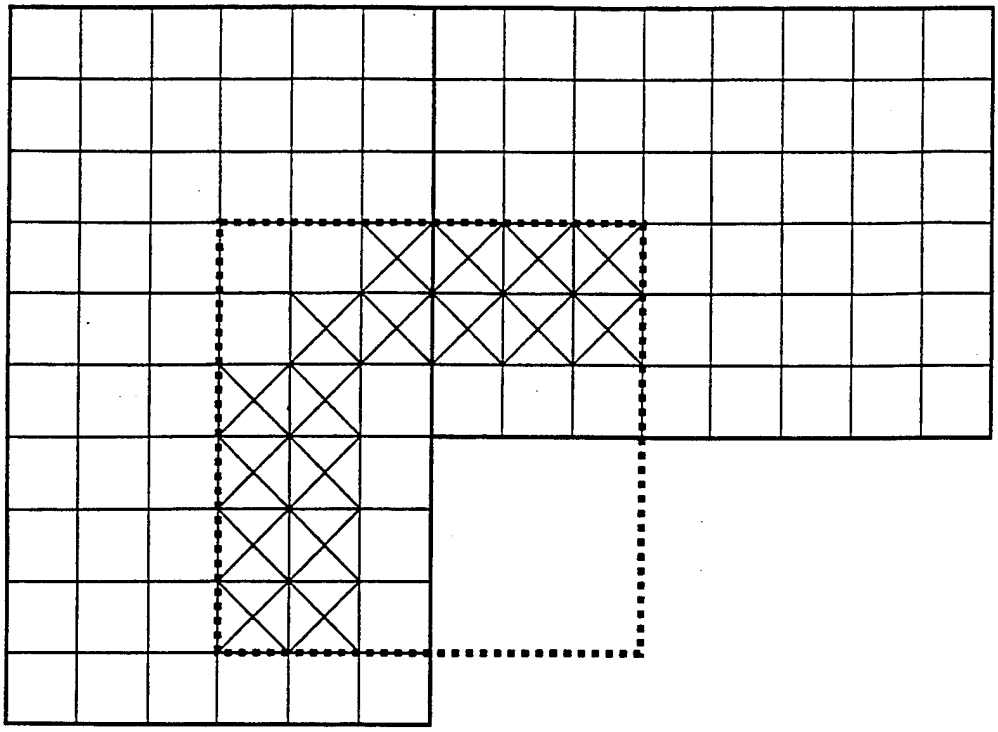


FIG. 3. *The role of the proper nesting list during regridding.*

above. Another purpose of the proper nesting list is to guide the selection of the patches on finer levels. The patches on the current level have been selected so that the interesting waves will lie inside the union of the patches until the patches have been advanced in time to synchronization with the next coarser level. We want to construct a list of patches on each of the finer levels, so that each fine patch lies inside the refinement of the proper nesting list belonging to its coarser level. When the union of patches on the current level is not convex, the proper nesting list prevents patches on the next finer level that straddle an interior corner of the union. As an example, Fig. 3 shows a collection of tagged cells (which are each marked with an X) on two patches (which are illustrated by heavy solid lines). The smallest box containing these tagged cells (which is illustrated with a heavy dotted line) lies partly outside the union of the patches. If this containing box were used to generate a refined patch, then some of the fine cells would not be properly nested.

The next regridding step is to update the list of patches on the finer level. First, cells in the patches belonging to the current level are tagged if their global integration error is too large. This procedure uses Richardson extrapolation to estimate the local truncation error in the integration, and a simple device to estimate the number of timesteps to be performed on this level of refinement. This error estimation procedure is a standard procedure in the numerical integration of ordinary differential equations [5]. Suppose that at each timestep we commit an error of magnitude ϵ (principally the local truncation error); further, suppose that the computation permits a bound M on the growth of these errors. Note that the conservative difference (3) shows that the computational solution essentially amounts to applying a perturbation of the identity operator to the previous solution; it is therefore reasonable to expect M to be close to 1 for smooth flow and sufficiently fine mesh. Then the error e_n at step n satisfies

$$e_1 <= \epsilon \,.\; e_n <= \epsilon + M e_{n-1} \text{ for } n > 1.$$

Then an argument by induction shows that

$$e_n \leq \epsilon \sum_{j=0}^{n-1} M^j = \epsilon \frac{M^n - 1}{M - 1}.$$

If M is close to 1, then for small n the error bound will be approximately $n\epsilon$. Suppose that the local truncation error satisfies

$$\epsilon = C\Delta t^{k+1},$$

where k is the expected global order of the scheme. (We are assuming that spatial and temporal error orders are equal.) Then the error in taking one coarse step of size $n\Delta t$ is

$$e_n^c \approx Cn^{k+1}\Delta t^{k+1}.$$

On the other hand, if we take n fine timesteps of size Δt, the error is

$$e_n^f \approx Cn\Delta t^{k+1}.$$

This allows us to estimate the local error of a fine timestep by

$$\epsilon \approx \frac{e_n^c - e_n^f}{n^{k+1} - n} = \frac{w_n^c - w_n^f}{n^{k+1} - n},$$

where w is the quantity being monitored for errors. This gives us a computable estimate for the local truncation error. The global error can be estimated by multiplying the local error by the anticipated number of timesteps N. Here,

$$N \approx \frac{L}{s\Delta t},$$

where L is some length scale associated with the problem, s is some important wavespeed, and Δt is the current timestep. Thus, cells are tagged if

$$\frac{|w_n^c - w_n^f|}{n^{k+1} - n} \frac{L}{s\Delta t} > \text{tolerance}.$$

If the regrid interval has the value k, the steps in the error estimation procedure are the following:

1. At the current level of refinement,
   (a) advance the data for one timestep Δt and
   (b) coarsen the results by a factor of k.
2. On a patch coarsened by a factor of k,
   (a) coarsen the data from the patch at the current time *minus* $(k-1)$ timesteps, and
   (b) advance the data for one timestep $k\Delta t$.
3. Compare the results of the two time integrations.

Note that the error estimation is performed on pseudopatches that potentially lie in index spaces between the current level of refinement and the next coarser level (since the pseudopatches are coarsened by a factor of the regrid interval, and mesh refinement uses an integer multiple of the regrid interval). This reduces the work in comparing the errors; in particular, we would not want to increase the work by integrating the fine patches and comparing them with the values on a coarser level, since the results on the fine patches would have to be discarded

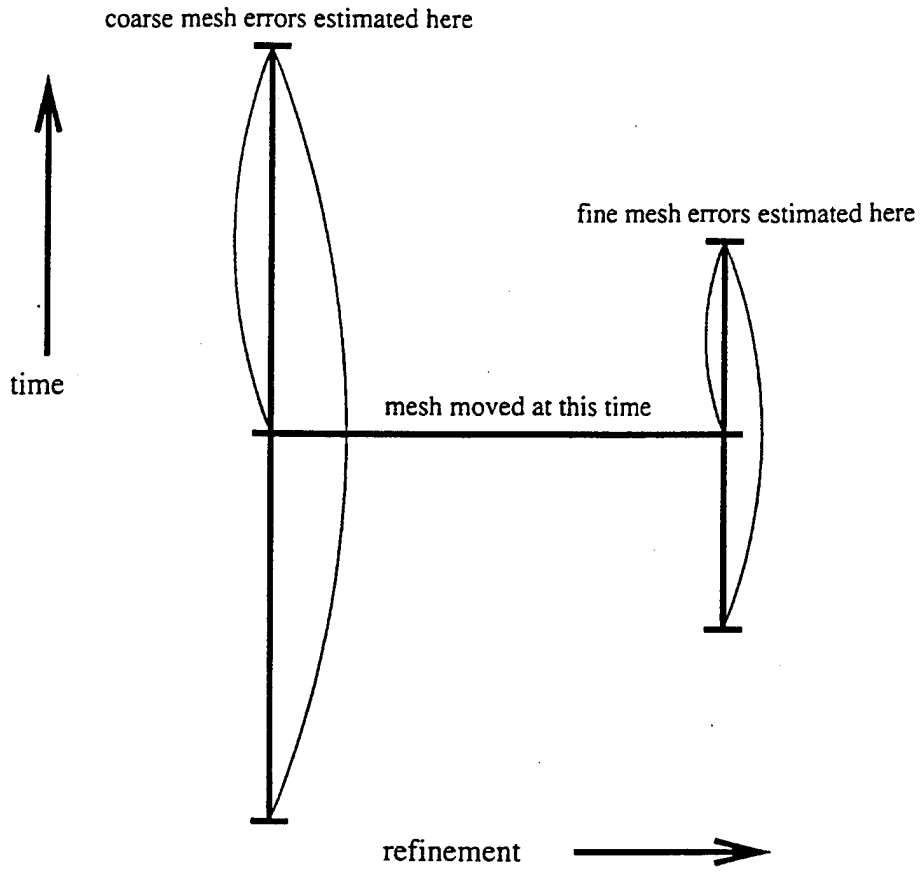


FIG. 4. *Timestepping for error estimation.*

when the patches are moved. The ordering of the coarsening and comparison operations also makes the algorithm simpler; if we were to compare errors on the current level of refinement by coarsening, integrating one coarse step, and then refining, we would have to construct a high-order conservative interpolation. In particular, this interpolation would have to be of a higher order than that used to construct predictor values for the numerical flux computation.

It is interesting to consider the implementation of this error estimation strategy on a recursively refined mesh. Figure 4 shows the timelines for two levels of refinement near a synchronization time; it shows how the initial data for the coarse and fine values are obtained for Richardson extrapolation. Note that errors on coarse and fine meshes are estimated at different times. At first glance, this would appear to be undesirable. However, the alternative of comparing errors on all levels at the same time actually leads to much wasted work and larger refined regions. This is because the error estimation on the coarse mesh places the refined cells where the disturbance will be moving, plus or minus buffer cells. Thus it is only necessary to buffer by regrid interval minus 1 cells, since that is the number of timesteps that will be taken between the times when the errors are estimated, and when the mesh will next be moved. If the errors had been computed at the same times, then it would be necessary to buffer by an additional cell on each level. Furthermore, the error estimation would have to proceed through more than one timestep, with recursive calls to integration on finer levels in order to provide data for finer grids. Since the mesh is going to be moved, this is extra work being performed for data that are mostly going to be discarded.

We want to prevent cells from being alternately coarsened and refined. If a cell is not currently tagged, it is tagged if its global error estimate exceeds some specified tolerance. On the other hand, if a cell is currently tagged, then it remains tagged unless its global error estimate falls below the specified tolerance divided by the regrid interval. This serves to prevent much of the chatter that occurs at the edges of refinements, where the error estimates hover around the error tolerance.

A cell is also tagged if it has an underlying fine cell. In order to determine these cells, we recursively update the patch lists on finer levels, and tag cells on the current level overlaying the tagged cells on the finer level.

Next, all cells sufficiently near the tagged cells are also tagged. The width of this buffer is called the "error buffer," and has been chosen to be equal to the regrid interval minus 1. This is the number of timesteps between the point where the errors were estimated and where the level will next be regridded.

The next step is to determine a list of boxes that contain the tagged cells. First, a large box containing all of the tags is found; then "cut points" are sought in order to split this big box into smaller boxes that cover the tagged cells more efficiently. A histogram of the cell tags in each coordinate direction helps us to determine how to cut the boxes. Cut points are selected according to goodness: a zero histogram entry near the center is best; an inflection point in the histogram near the center is next best; the midpoint is the choice of last resort. The best of these overall coordinate directions is chosen. This procedure is due to Berger, and differs from that in [2].

After this initial list of boxes is determined, it is further massaged. First, each box is further subdivided, if necessary, so that it lies inside the proper nesting list. Next, if the edge of a box falls too near a physical boundary, it is extended all the way to the boundary. Afterward, the list of boxes is searched to see if any two share a common side, so that they can be coalesced to make a larger box. Boxes that are too large (i.e., they require more temporary space than we are willing to provide) are subdivided. Once these boxes are found, they are shrunk to the smallest size needed to contain the tagged cells, then expanded within their former boundaries to avoid physical boundaries among the ghost cells.

The final step is to make the new patches. If there are no tagged cells, then the current finer level is destroyed if it exists. If there are tagged cells and no current finer level, then a new finer level is created; the data on these new fine patches are obtained by refining the data on the overlying patches in the current level of refinement. If there are tagged cells and a finer level already exists, then the data on the new fine patches are determined by copying the data from the old fine patches where they are available and refining them from the current level otherwise. In this case, a temporary memory bulge can occur while the finest level data are copied from old patches to new.

3.5. Refluxing. Numerical integration on an adaptively refined mesh involves communication between coarse and fine mesh patches. We have already seen that boundary data for fine patches may be obtained from refinement of data on coarser patches. Conversely, fine patches produce more accurate results that can be used to correct the data on coarser patches.

This coarsening of data takes two forms. First, when a coarse cell overlies finer cells, the data on the fine cells are coarsened and replace the coarse data. The computational form of this coarsening procedure varies from one flow variable to another and will be discussed in §4. However, we remark that the coarsening of conserved quantities needs to be conservative: that is, the volume-weighted average of the conserved variables on the fine grid replaces the conserved variables on the overlying coarse cells. This coarsening process, by itself, could not preserve conservation; it is also necessary to replace the coarse fluxes around the boundary of the fine patches with the boundary and time integral of the fluxes determined on the fine

patches. In this way, the change in the values of conserved quantities in cells overlying finer patches is compensated by changes in conserved quantities on cells neighboring the refinement.

Because of the peculiarities of the hypoelastic equations of state for solids, we perform this refluxing process in a form different from that in [2]. We compute the time integrals of the fine fluxes, then coarsen them in space to provide improved values of the coarse fluxes; afterward, we repeat the conservative difference (3). This amounts to more work than the equivalent process in [2]. However, other solid mechanics variables (such as stress) may have a *nonlinear* dependence on the fluxes and need recomputation. This is discussed in more detail in §4.

3.6. Initialization. The beginning of the computation is somewhat different from the process presented above. On the coarsest level, the patches are determined by subdividing the physical domain into pieces that are not too big for the integrator; on finer levels, the patches are assumed to be determined by the initialization process on the next coarser level. Then, a user-supplied procedure is called to place the initial data on the patches; this procedure also determines a stable timestep.

If the current level can be refined, then a special error estimation process is conducted. Here, the initial data are advanced regrid interval timesteps and coarsened; they are also coarsened and advanced one timestep. The two results are compared in order to estimate the global truncation errors, and cells with unacceptably large errors are tagged. These cells are buffered by regrid interval cells in each coordinate direction, since that is the number of timesteps that will be taken before the first regridding is performed.

The tagged cells are organized into patches on a new fine level just as described above. Then the user-supplied procedure is called to determine the initial data on the new fine patches. If the new fine level can be refined, we advance it forward one timestep in order to provide boundary data for the recursive initialization process on finer levels. After the finer levels have been initialized, we return to the initial data, forgetting the results of the integrations on all of the levels, since the size of the first timestep on each level may be affected by the work on coarser levels during the normal integration process.

4. Modifications for solid mechanics. Adaptive mesh refinement for gas dynamics tacitly assumes that all of the important variables are either conserved or computed directly from the conserved quantities. As a result, all of the flow variables are cell-centered, except for the fluxes. Furthermore, the conserved flow variables are coarsened by conservative averaging and refined by conservative interpolation.

The situation is more complicated in solid mechanics. The principal difference lies in the variety and complexity of the kinetic equations of state. For example, in hyperelastic equations of state, the stress is computed at cell centers directly from the deformation gradient. On the other hand, elastic-plastic models usually describe the stress by a constrained ordinary differential equation involving the velocity gradient and material history as input. Stress integrals are computed along particle paths, which fortunately stay fixed in space on Lagrangian grids; in our algorithm, these particle paths are the cell centers. For such variables, conservative averaging is an inappropriate coarsening process; rather, the selection of a particular fine particle path for coarse data is more reasonable. (See Fig. 5; there, fine particle paths are denoted by circles, and coarse particle paths are marked by squares.) In other words, the coarsened data for these variables are taken to be the fine level data on the particle paths shared by the coarse and fine patches. Refinement of stress data might also be complicated. It may be desirable to guarantee that refinement of a stress tensor constrained by a yield surface is also at yield. In such a case, some refinement of the history parameters will determine the constraints that need to be enforced on the stress during refinement. It is clear that this process

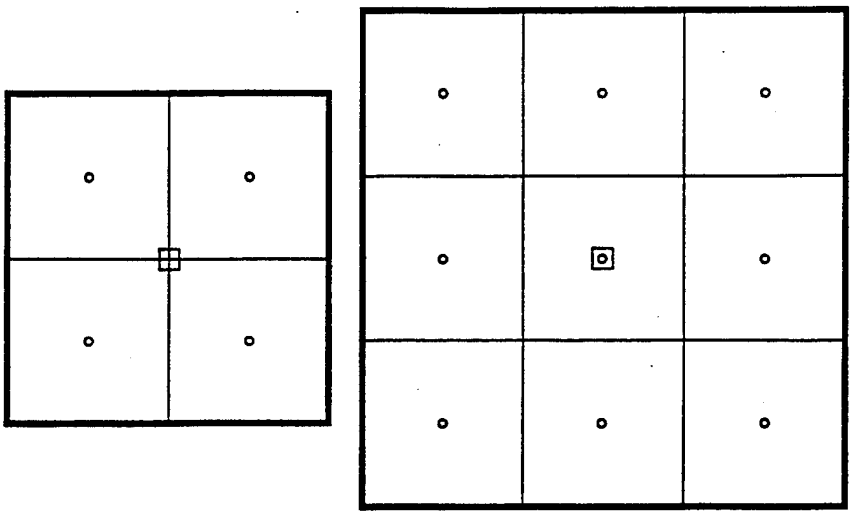


FIG. 5. *Coarse and fine particle paths with even and odd refinement ratios.*

is model dependent and must not be made part of the organizational structure of adaptive mesh refinement.

In order to allow for coarsening along particle paths, it is necessary to restrict the values of the refinement ratio. When a cell-centered variable (such as a hypoelastic stress) coarsens along particle paths, the refinement ratio needs to be odd; if no cell-centered variable coarsens along particle paths, then the refinement ratio can be even, as in [2]. The regrid interval is the smallest allowable refinement ratio. We take it to be 3 if some variable coarsens along particle paths; otherwise it is 2.

We have seen [18] that the accuracy of integration of elastic-plastic equations of state is the determining factor in the overall accuracy of the motion. However, timestep selection in Godunov methods is normally based on the CFL condition for the conservation law; this might lead to a timestep that is too large for acceptable accuracy in the stress integration. Adaptive mesh refinement allows the accuracy of the kinetic equation of state to influence the timestep size. By measuring the errors between coarse and fine integrations of the stress (say, errors in pressure and second invariant of deviatoric stress), it is possible to force both mesh and time refinement where the equation of state is inaccurate. Of course, it is crucial that the equation of state be integrated with the same order of truncation error as the conservation laws; for a discussion of the difficulties involved in integrating elastic-plastic models, see [14].

Note that the use of 3 as a regrid interval can cause some minor complications. When the regrid interval is 2, the discussion in §3.4 shows that the difference between coarse and fine values is divided by 6 ($= 2^3 - 2$) to estimate the local truncation error; when the regrid interval is 3, the difference is divided by 24 ($= 3^3 - 3$). If the asymptotic estimates of the errors are not yet valid because the mesh is too coarse, the division by 24 makes errors appear to be smaller than they would with a regrid interval of 2; this could prevent refinement on a coarse mesh with a loose error tolerance. Also note that a regrid interval of 2 allows refinement ratios of 2, 4, 6, 8, ...; a regrid interval of 3 allows refinement ratios of 3, 9, 15, 21, ... Thus when the regrid interval is 3, the selection of a refinement ratio larger than the regrid interval can have the effect of concentrating the work on the finest level much more intensely. Further, cells tagged for errors in refinement need to be buffered by an extra cell when the regrid interval is 3; this leads to somewhat larger patches, and somewhat less frequent regridding. The effect on the overall cost of the algorithm is still being investigated.

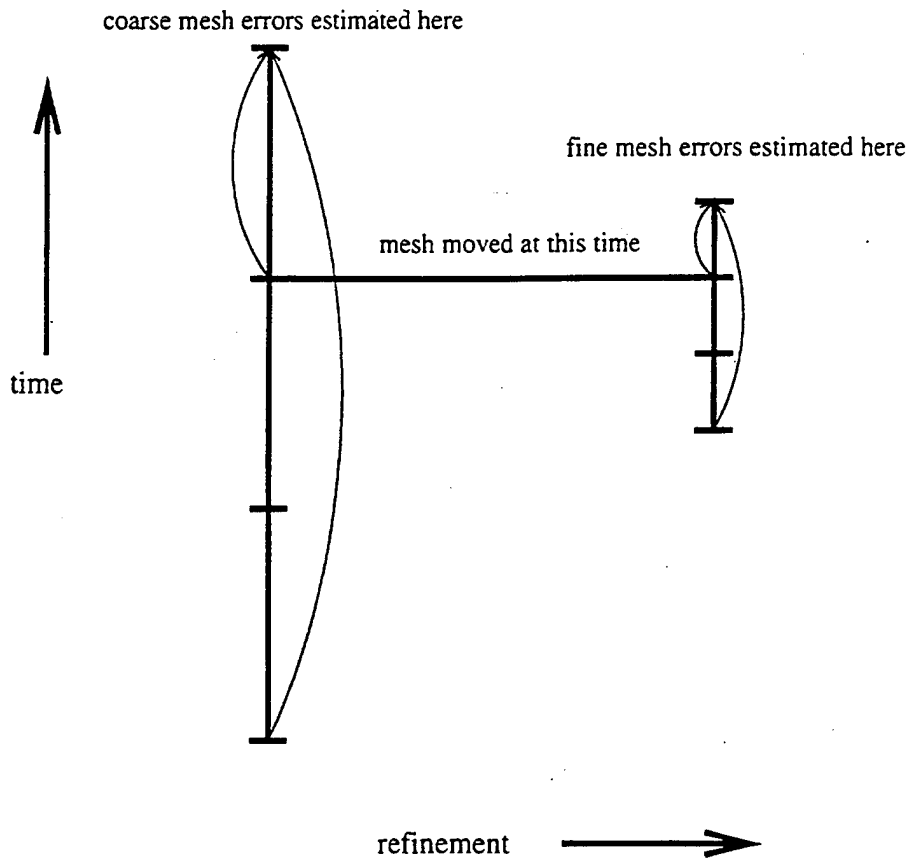


FIG. 6. *Timestepping for error estimation with an odd refinement ratio.*

A more interesting effect of the choice of 3 as a regrid interval is the effect on data storage for error estimation. Figure 4 shows the timelines for error estimation with a regrid interval of 2. Note that it is necessary to retain data at both the current time and one previous time in order to advance both the patch data and its coarsened values up to the next time. When the regrid interval is 3, it is necessary to store 3 timelevels of data (see Fig. 6). However, this storage is not needed on the finest level, since error estimation is not performed on that level. Thus, this change does not add significantly to the storage requirements for the algorithm.

5. **Object-oriented programming.** As we mentioned in the Introduction and demonstrated in the sections that followed, adaptive mesh refinement involves much more complicated programming than straightforward integration on regular meshes. This program complexity is reflected not only in the data structures used to represent the patches on various levels of refinement, but also in the communication between the patches.

Other aspects of solid mechanics computations also add to the program complexity. It is common for problems to involve multiple materials with different equations of state. Oftentimes the computations in the equations of state involve a large number of temporary variables, which could lead to large memory requirements if not handled carefully. Users also need to perform computations in one, two, or three dimensions.

It was our goal to implement adaptive mesh refinement so that the mesh hierarchy and patch communication were independent of the equation of state and the number of physical

dimensions. We also wanted the treatment of various kinds of flow variables to be driven by requests from the equation of state, with no modification of the adaptive mesh refinement program structure involved in changes in the equation of state.

Current languages, such as FORTRAN 77, which are highly efficient for computations on rectangular arrays of data, do not provide the variety of data or programming structures that would make the implementation of adaptive mesh refinement easy, either for implementation or maintenance. FORTRAN 90 was not available when we began this work, and still does not offer all the features we need. As a result, we decided to implement the adaptive mesh refinement program structure in C^{++}, with the numerically intensive routines written in FORTRAN.

There are three main abstract structures in our implementation of adaptive mesh refinement: a BOX, a FLOW-VARIABLE, and an EQUATION-OF-STATE. We will describe each of these in turn.

In one-dimension a BOX is a interval in index space; in two dimensions it is a rectangle of indices; in three dimensions its name is descriptive. This data structure involves two vectors of indices for the first and last corners of the BOX. A BOX is very basic to adaptive mesh refinement, because every patch of any level of refinement is a BOX endowed with extra data. All of the basic adaptive mesh refinement communication between patches (the determination of boundary data, coarsening, and refinement, as well as error estimation) involves operations with BOXes. For example, regridding is the process of determining a new list of BOXes to cover tagged indices of cells in some given list of BOXes. When adaptive mesh refinement is expressed in terms of basic operations on an abstract BOX, the program structure can become dimensionally independent. These basic operations involve actions such as determining if a BOX contains a given cell index, finding the intersection of two BOXes, refining and coarsening, and growing by some number of cells in the various coordinate directions.

In C^{++} and other object-oriented languages, the operations defined on abstractions such as the BOX class are available to other derived classes. Thus, we have implemented TAGBOXes for the handling of error estimation and other related operations; its additional data basically consist of an array of BOOLEAN values for each cell in the BOX. The PATCH class is derived from a TAGBOX and contains extra information for the current state of the solution to the equations of motion. Both TAGBOXes and PATCHes can perform the BOX operations on other BOXes.

A FLOW-VARIABLE describes the information needed by adaptive mesh refinement in order to perform interpatch communication. This information consists of the input-output status, the geometry (such as location at cell centers or corners), the number of variables, and the variable name. REAL-FLOW-VARIABLEs are FLOW-VARIABLEs with descriptions of their desired schemes for refinement and coarsening between PATCHes on different levels; they also remember pointers to the data at various stages of the computation. Lists of REAL-FLOW-VARIABLEs can determine the total amount of data required to store everything with a specific iostatus and communicate this to PATCHes. In this way, the PATCHes can distinguish between old and new data, particularly when it is time for error estimation and regridding.

An EQUATION-OF-STATE is the problem-specific part of the code. It consists of lists of FLOW-VARIABLEs and procedures to act on the FLOW-VARIABLEs. When it is time to advance the data on a PATCH, the PATCH uses its pointer to the EQUATION-OF-STATE to invoke *run-time binding* of the integration techniques appropriate to the material model. Thus the adaptive mesh refinement code does not need to know anything about the integration techniques, the tests for error estimation, the determination of the variables intended for display, or even the FLOW-VARIABLEs themselves. Whenever these operations are needed,

each PATCH uses its pointer to the EQUATION-OF-STATE to invoke the correct procedures. This means that the user can insert new equations of state by making them known to the abstract EQUATION-OF-STATE class in only two places. The first is where the name of the specific EQUATION-OF-STATE (such as MOONEY-RIVLIN or LINEAR-ELASTICITY) is read by the input data just before being constructed; the second is where the problem model names are read from restart files for a similar purpose.

As a final comment, we would like to emphasize that object-oriented programming does not make the initial programming task less time consuming. Because of the data protection mechanisms in C^{++}, it is necessary to write a variety of simple access procedures to obtain certain information safely; this seems to increase the size of the code as well. On the other hand, object-oriented programming does seem to make program modification much quicker and more trouble-free. This advantage outweighs the other two disadvantages.

6. Numerical results. We will present two numerical examples. The first problem is Lamb's problem [1] for a line source on the surface of a linearly elastic solid. We have chosen the material to have a Poisson ration of 0.25. The calculation was performed using four levels of mesh (three levels of refinement) on an initial 12×12 coarse grid, and with a global error tolerance of 0.1. Since the equation of state was written in hypoelastic form, using equations for the stress rate at the cell centers, we used a refinement ratio of 3. Thus the finest level was equivalent to a 324×324 mesh.

In this problem, the top boundary is stress-free, except for a specified normal force applied for one timestep on the finest mesh at the center of the top side of the cell in the top left-hand corner. The left boundary is a line of symmetry, and the bottom and right boundaries are transmitting boundaries. The numerical treatment of the symmetry boundary takes the velocity to be odd in the normal direction and even in the tangential direction; this implies that the stress tensor is even on the diagonal and odd off the diagonal. The transmitting boundaries are easily treated by setting the Godunov slopes to 0 in the cells next to the transmitting boundaries, and choosing the solution of the Riemann problem to be the state on the inside of the domain. The treatment of the free surface sets the slopes for the vertical direction in the row of cells to be the same as in the cells just below them. The rest of the computation at the free surface acts as if there were a 0-strength material just above the surface.

The contours of the second invariant of the deviatoric stress are shown in Fig. 7. This figure also shows the boundary of the patches at the different levels of refinement. Figures 8–9 show the adaptive refinement of the grid at various times in the simulation. The careful reader will notice that at late time the mesh is not coarsening to levels 1 or 0 anywhere in the problem. This is due to two factors. The first is that using a regrid interval of 3 requires a wider buffer around cells tagged for refinement, compared with a regrid interval of 2. The second cause is that the error estimation is done on a coarsening of the level; with a regrid interval of 3 instead of 2 this means that a individual cell with unacceptable error causes more cells to be tagged for refinement. Thus, there is some additional cost associated with allowing variables to coarsen along particle paths. The hope is that the extra cost is compensated by increased accuracy in the integration of the stress rate equations, but this hypothesis needs to be tested on more complicated models than linear elasticity. Also notice that there is a very thin patch just to the right of the center of the problem that overlaps its neighbor. While the regridding process attempts to avoid this, it is difficult to prevent. In this case, the pattern-recognition process attempted to create a very thin patch. If the patch had been located next to a reflecting wall, there would not be enough interior cells to provide the required boundary data. To eliminate this problem, all patches are prevented from being too thin.

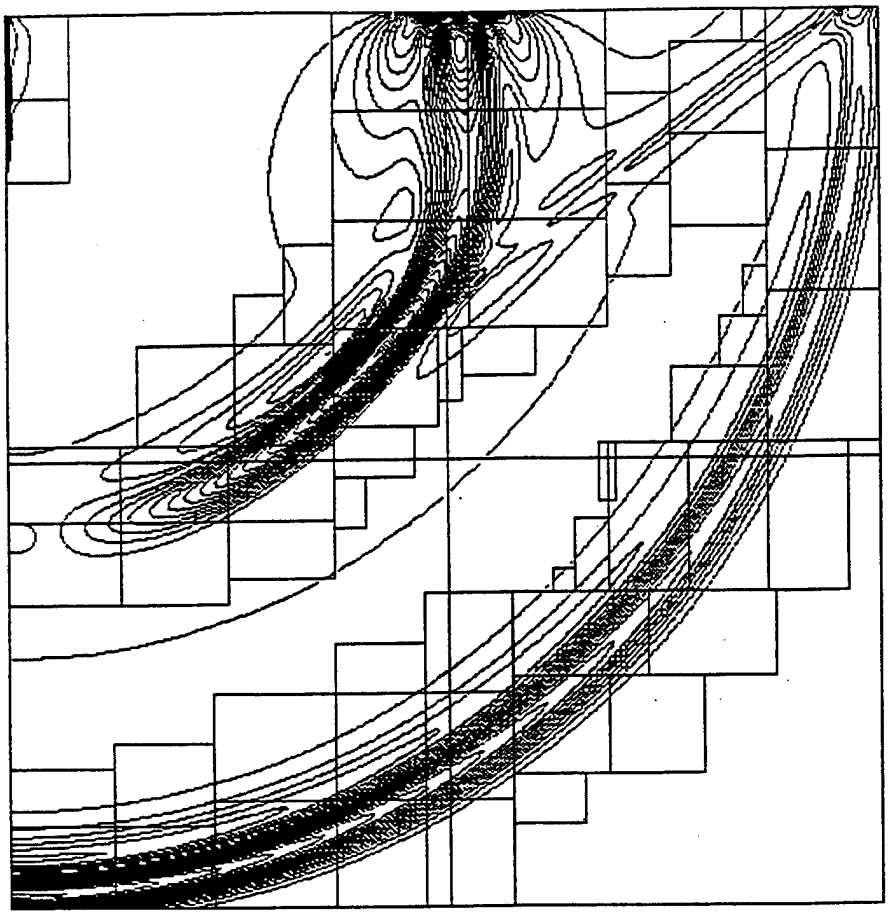


FIG. 7. *Contours of second invariant of deviatoric stress for Lamb's problem.*

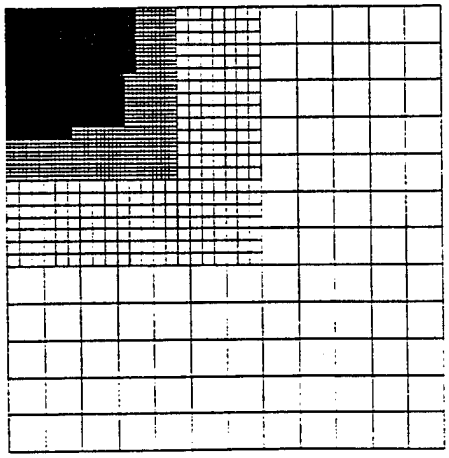


FIG. 8. *Adaptive mesh for Lamb's problem: early time.*

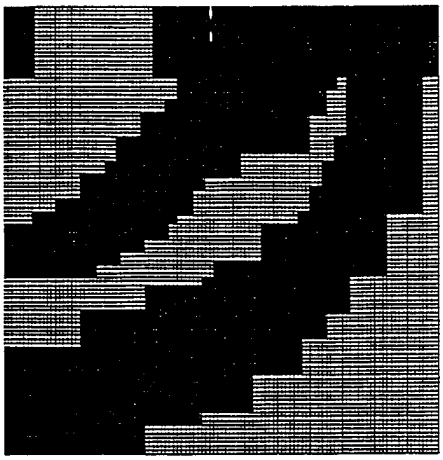


FIG. 9. *Adaptive mesh for Lamb's problem: late time.*

TABLE 1
Computational time for Lamb's problem using adaptive and uniform meshes (in seconds on an SGI Indigo R4000).

Simulation time (msec.)	Computation time (sec.)					
	Level 0	Level 1	Level 2	Level 3	Total	Regridding
54	11.15 859.89	0.81	3.16	19.73	34.85 859.89	2.28
109	42.28 1685.85	2.99	14.19	108.46	167.92 1685.85	13.61
163	84.45 2494.15	7.68	35.53	317.24	444.6 2494.15	33.84
217	137.72 3298.85	14.04	69.13	676.31	897.2 3298.85	66.40
272	187.56 4166.67	21.31	108.01	1132.92	1449.8 4166.67	100.91

In Table 1 the timing comparison between the adaptive mesh refinement algorithm and a fixed mesh computation shows some interesting results. Between 7 and 8% of the simulation time is spent in computational overhead for error estimation, refluxing, and regridding: this is comparable to the overhead reported for the gas dynamics algorithm [2]. However, the computational examples were run until the waves of interest covered a large portion of the computational domain. As a result, for a large portion of the calculational time the adaptive mesh refinement algorithm was using much closer to the same amount of mesh as a fixed grid calculation. In this particular calculation, which was run to a point at which much of the problem domain is refined, adaptive mesh refinement was about 2.87 times faster than a fixed fine mesh. Even though this is significantly less speedup than reported by the gas dynamics adaptive mesh refinement algorithm for Mach reflection problems, it still translates into several hours of savings on a mega-flop workstation. A careful reader will also notice that the computation time on Level 0 is significantly less during the first 54 msec of simulation than in any later 54 msec. This is due to the smaller amount of refluxing and coarsening performed when the disturbance occupies a small portion of the problem domain.

For our second numerical example, we present a calculation of a numerical "hammer hit" to a nonlinear elastic solid. In this case, over a finite interval and a finite amount of time force is applied to a nonlinear solid described by the Mooney–Rivlin model. The bulk and

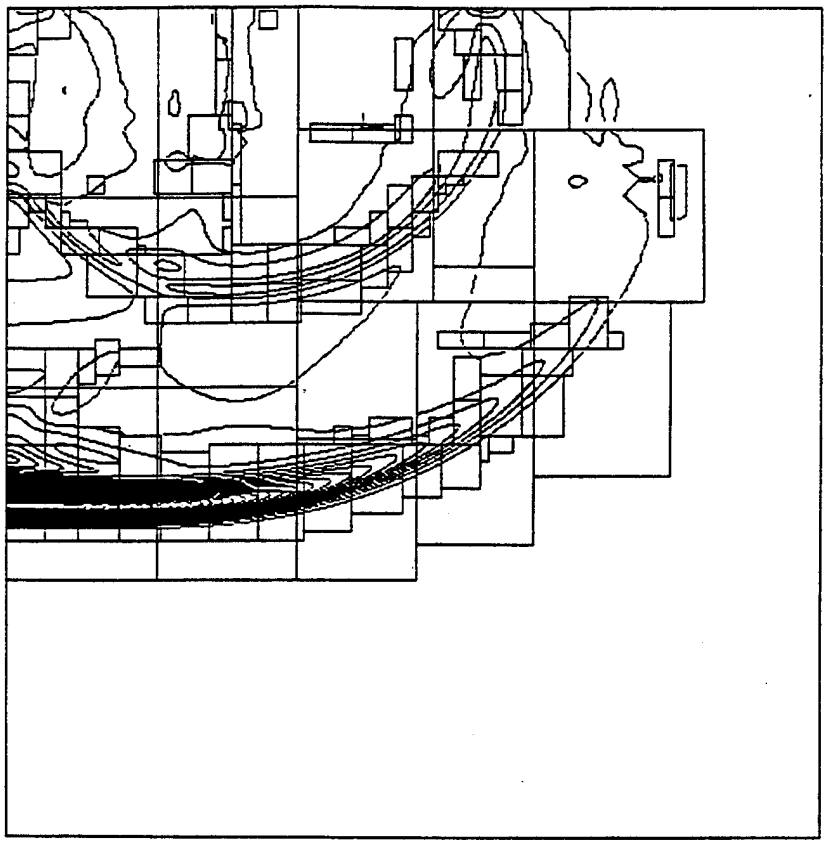


FIG. 10. *Contours of deviatoric stress in hammer hit.*

shear moduli are the same as in Lamb's problem, namely, $\kappa = 1$ and $\mu = .6$, leading to a Poisson ratio of $\nu = 0.25$. This calculation is similar to Lamb's problem, in that it involves a semi-infinite solid undergoing a line load in plane strain, with a symmetry boundary at the left, nonreflecting boundaries at the bottom and right, and a top surface that is stress-free except for a 20-unit force applied to the right 25% of the surface until time $1. \times 10^{-3}$. During this time, the stress varies linearly in space to zero value at 30% of the distance from the left. Since this model is hypoelastic, there is no need to compute stress histories along particle paths, and it is possible to use a mesh refinement ratio of 4. Our initial mesh was 12×12, and four levels of mesh (three levels of refinement) were allowed, leading to a finest mesh equivalent to 768×768. The initial timestep was 5×10^{-3} seconds on the coarsest mesh, which was small compared with the CFL timestep of 5.6×10^{-2} for the material at rest. The small initial timestep was necessary to allow for the increase in the characteristic speed as the material was compressed and stiffened. During the calculation, the timestep was allowed to increase by a factor of 1.1 if the local CFL analysis allowed.

The norm of the deviatoric stress is plotted in the Eulerian frame in Fig. 10. Although the initial deformation of the surface of the solid is no longer evident, there is still significant indication of the waves that form between the initial compression and the free surface. In Figs. 11 and 12 we also show the mesh at early and late times in order to illustrate the grid adaptation.

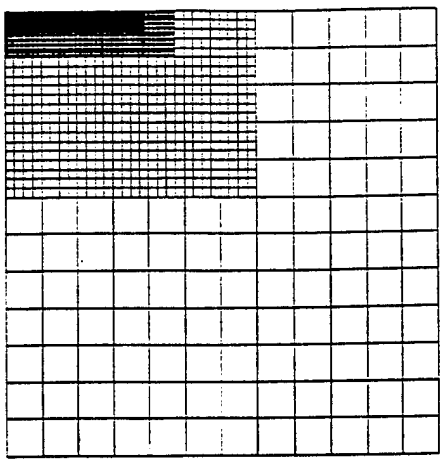


FIG. 11. *Adaptive mesh in hammer hit.*

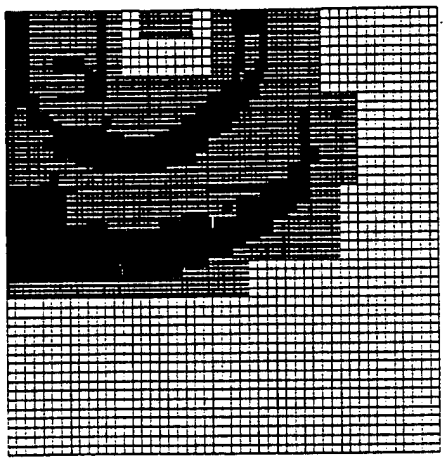


FIG. 12. *Adaptive mesh in hammer hit.*

The calculation took a little over five hours to reach the point at which the stress contours are shown. We attempted to run the calculation with a uniformly fine mesh (768×768 or 589824 cells), but the memory requirements exceeded the available swap space on the machine. This, of course, points out one of the added benefits of adaptive mesh refinement: the ability to run larger problems than would be possible otherwise.

7. Summary. We have extended the two-dimensional adaptive mesh refinement of Berger and Colella from gas dynamics to solid dynamics. This extension involved several modifications to the algorithm, including the use of even or odd refinement ratios and more elaborate refluxing steps. Object-oriented programming was used to separate the model-specific portions of the algorithm from adaptive mesh refinement, to develop the adaptive mesh refinement in a dimensionally independent form, and to allow for flexible and reusable data structures. The resulting algorithm produces significant computational savings with an overhead cost slightly less than 10% for error estimation, refluxing, and regridding.

Acknowledgments. We thank John Bell, Bill Crutchfield, and Mike Welcome of Lawrence Livermore National Laboratory for many thoughtful conversations during the course of this work.

REFERENCES

[1] J. D. ACHENBACH, ED., *Wave Propagation in Elastic Solids*, North–Holland, Amsterdam, 1973.

[2] M. J. BERGER AND P. COLELLA, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comp. Phys., 82:64–84, 1989.

[3] P. COLELLA, *Multidimensional unsplit methods for hyperbolic conservation laws*, J. Comp. Phys., 87:171–200, 1990.

[4] N. CRISTESCU, *Dynamic Plasticity*, North–Holland, Amsterdam, 1967.

[5] G. DAHLQUIST AND Å. BJÖRCK, *Numerical Methods*, Translated by N. Anderson, Prentice–Hall, Englewood Cliffs, NJ, 1974.

[6] C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice–Hall, Englewood Cliffs, NJ, 1971.

[7] B. L. KEYFITZ AND H. C. KRANZER, *A system of non-strictly hyperbolic conservation laws arising in elasticity theory*, Arch. Rat. Mech. Anal., 72:220–241, 1980.

[8] J. D. LAMBERT, *Numerical Methods for Ordinary Differential Systems*, John Wiley & Sons, New York, 1991.

[9] R. LÖHNER, K. MORGAN, J. PERAIRE, AND M. VAHDATI, *Finite element flux-correct transport (FEM-FCT) for the Euler and Navier–Stokes equations*, Internat. J. Numer. Methods Fluids, 7:1093–1109, 1987.

[10] R. N. MILLER AND K. MILLER, *Moving finite elements I*, SIAM J. Numer. Anal., 18:1019–1032, 1981.

[11] M. SHEARER, *The Riemann problem for a class of conservation laws of mixed type*, J. Differential Equations, 46:426–443, 1982.

[12] J. C. SIMO AND M. ORTIZ, *A unified approach to finite deformation elastoplasticity based on the use of hyperelastic constitutive equations*, Comput. Methods Appl. Mech. Engrg., 49:221–245, 1985.

[13] A. TANG AND T. C. T. TING, *Wave curves for the Riemann problem of plane waves in isotropic elastic solids*, Internat. J. Engrg. Sci., 25:1343–1381, 1987.

[14] J. A. TRANGENSTEIN, *A second-order algorithm for the dynamic response of soils*, Impact Comput. Sci. Engrg., 2:1–39, 1990.

[15] ———, *A comparison of two numerical methods for shocks in one-dimensional elastic-plastic solids*, in Viscous Profiles and Numerical Methods for Shock Waves, M. Shearer, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991.

[16] ———, *A second-order Godunov algorithm for two-dimensional solid mechanics*, Comput. Mech., 13:343–359, 1994.

[17] J. A. TRANGENSTEIN AND R. B. PEMBER, *The Riemann problem for longitudinal motion in an elastic-plastic bar*, SIAM J. Sci. Statist. Comput., 12:180–207, 1991.

[18] ———, *Numerical algorithms for strong discontinuities in elastic-plastic solids*, J. Comp. Phys., 103:63–89, 1992.

[19] P. WOODWARD AND P. COLELLA, *The numerical simulation of two-dimensional fluid flow with strong shocks*, J. Comp. Phys., 54:115–173, 1984.

DISTRIBUTION LIST

DNA-TR-94-143

DEPARTMENT OF DEFENSE

DEFENSE INTELLIGENCE AGENCY
ATTN: DIW-4
ATTN: RTS

DEFENSE NUCLEAR AGENCY
ATTN: SPWE
2 CY ATTN: SSTL

DEFENSE TECHNICAL INFORMATION CENTER
2 CY ATTN: DTIC/OCP

FIELD COMMAND DEFENSE NUCLEAR AGENCY
ATTN: FCPR
ATTN: FCT COL H LAWSON

FIELD COMMAND DEFENSE NUCLEAR AGENCY
ATTN: FCTO

DEPARTMENT OF THE ARMY

ARMY RESEARCH LABORATORIES
ATTN: AMSRL-WT-NJ

DEP CH OF STAFF FOR OPS & PLANS
ATTN: DAMO-NCZ

PED MISSILE DEFENSE SFAE-MD-TSD
ATTN: ATC-D WATTS
ATTN: CSSD-H-SA

U S ARMY SPACE STRATEGIC DEFENSE CMD
ATTN: CSSD-SA-E

DEPARTMENT OF THE NAVY

NAVAL RESEARCH LABORATORY
ATTN: CODE 5227

OFFICE OF CHIEF NAVAL OPERATIONS
ATTN: NUC AFFAIRS & INT'L NEGOT BR

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY LIBRARY
ATTN: AUL-LSE

WL/ELE BLDG 620
ATTN: WRDC/MLP
ATTN: WRDC/MTPM

DEPARTMENT OF ENERGY

LAWRENCE LIVERMORE NATIONAL LAB
ATTN: J CAROTHERS
ATTN: G GOUDREAU
ATTN: P CHRZANOWSKI
ATTN: DR S J SACKETT
ATTN: G POMYKAL

SANDIA NATIONAL LABORATORIES
ATTN: 9 J RIGALI 9800

OTHER GOVERNMENT

CENTRAL INTELLIGENCE AGENCY
ATTN: OSWR/NED 5S09 NHB

DEPARTMENT OF THE INTERIOR
ATTN: D RODDY

DEPARTMENT OF DEFENSE CONTRACTORS

AEROSPACE CORP
ATTN: H MIRELS

DUKE UNIVERSITY
2 CY ATTN: J A TRANGENSTEIN

INFORMATION SCIENCE, INC
ATTN: W DUDZIAK

JAYCOR
ATTN: CYRUS P KNOWLES

KAMAN SCIENCES CORP
ATTN: D CAYNE

KAMAN SCIENCES CORP
ATTN: DASIAC

KAMAN SCIENCES CORPORATION
ATTN: DASIAC

MAXWELL LABORATORIES, INC
ATTN: J MURPHY

PACIFIC-SIERRA RESEARCH CORP
ATTN: H BRODE

SCIENCE APPLICATIONS INTL CORP
ATTN: H WILSON

SCIENCE APPLICATIONS INTL CORP
ATTN: DIV 411 R WESTERFELDT

SCIENCE APPLICATIONS INTL CORP
ATTN: J COCKAYNE

SCIENCE APPLICATIONS INTL CORP
ATTN: G BINNINGER

TRW INC
ATTN: TIC

TRW SPACE & DEFENSE SECTOR
ATTN: D M LAYTON
ATTN: W WAMPLER

VITRO CORP
ATTN: H BRIGHT

WEIDLINGER ASSOCIATES, INC
ATTN: P WEIDLINGER